지배소 후위 제약을 적용한 트랜지션 시스템 기반 한국어 의존 파싱 모델

임준호, 윤여찬, 배용진, 임수종, 김현기, 이규철O
한국전자통신연구원 지식마이닝연구실, 충남대학교 컴퓨터공학과O
{joonho.lim, ycyoon, yongjin, isj, hkk}@etri.re.kr, kclee@cnu.ac.krO

Korean Dependency Parsing Model based on Transition System using Head Final Constraint

Joon-Ho Lim, Yeo-Chan Yoon, Yongjin Bae, Hyunki Kim, Kyu-Chul LeeO
Electronics and Telecommunications Research Institute, Chung-nam UniversityO

요 약

한국어 의존 파싱은 문장 내 단어의 지배소를 찾음으로써 문장의 구조적 중의성을 해소하는 작업이다. 지배소 후위 원칙은 단어의 지배소는 자기 자신보다 뒤에 위치한다는 원리로, 한국어 구문분석을 위하여 널리 사용되는 원리이다. 본 연구에서는 한국어 지배소 후위 원리를 의존 파싱을 위한 트랜지션 시스템의 제약 조건으로 적용하여 2가지 트랜지션 시스템을 제안한다. 제안 모델은 기존 트랜지션 시스템 중 널리 사용되는 arc-standard와 arc-eager 알고리즘에 지배소 후위 제약을 적용한 포워드(forward) 기반 트랜지션 시스템과, 트랜지션 시스템의 단점인 에러 전파(error propagation)를 완화시키기 위하여 arc-eager 알고리즘의 lazy-reduce 방식을 적용한 백워드(backward) 기반 트랜지션 시스템이다. 실험은 세종 구구조 말뭉치를 의존구조로 변환하여 실험하였고, 실험 결과 백워드 기반 트랜지션 시스템이 포워드 방식보다 우수한 성능을 보였다. 기존 연구와의 비교를 위하여 기존 연구를 조사하였지만 세부 실험 환경이 서로 달라서 직접적인 비교는 어려웠다. 제안하는 시스템의 최고 성능은 UAS 92.85%, LAS 90.82% 이다.

주제어: 지배소 후위 제약, 트랜지션 시스템 기반 파싱, 한국어 의존 파싱

1. 서론

구문분석은 문장의 구조적 중의성을 해소하는 작업이다. 이는 자연어로부터 정보와 지식을 획득하기 위한 기초적인 작업으로 다양한 응용 분야에 활용된다. 구체적으로 문장의 의미 분석을 위한 의미역(semantic role label) 할당, 이벤트 및 시간/공간 정보 추출, 지식 추출을 통한 온톨로지의 확장, 질의응답(Question Answering)과 같은 응용 분야에 활용된다.

최근 많이 연구되고 있는 구문분석 기술은 의존 구문분석으로, 이는 문장의 각 단어에 대해서 지배소를 찾음으로써 구조 중의성을 해소한다. 의존 구문분석은 특히 2000년대 중반 CoNLL Shared Task를 통하여 많은 연구가 이뤄졌다 [1].

최근 의존 구문분석의 주된 방법론으로 그래프 기반 접근 방법과 트랜지션 기반 접근 방법이 있다. 그래프 기반 접근 방법은 문장의 각 단어를 그래프의 노드로 보고, 의존 관계를 엣지로 바라본 후, 그래프의 전체 점수를 최고로 하는 최대 스패닝 트리(Maximum Spanning Tree)를 구하는 문제로 구분분석을 바라본다 [2]. 이는 그래프 전체에 대해서 최적의 해를 구하기 때문에, 높은 구문분석 성능을 보이지만, $O(n^3)$ 이상의 높은 시간 복잡도를 가진다는 단점이 있다.

반면, 트랜지션 기반 접근방법은 입력 문장에 대해서 오토마타와 같은 트랜지션 시스템을 구동하는 방식으로, 매 단계마다 아크 연결, 단어 시프트 등의 연산을 순차적으로 결정하여 파싱을 수행한다 [3]. 트랜지션 기반 접근방법은 CoNLL Shared Task에서 그래프 기반 접근 방법과 비슷한 수준의 높은 성능을 보였다. 그리고 시간 복잡도가 O(n)으로 매우 빠른 성능을 보인다는 장점을 가진다. 하지만, 한 번 잘못된 결정을 하면 에러가 전파(error propagation)된다는 단점이 있다.

본 논문에서는 질의 응답과 같은 실시간 시스템에서 활용이 가능한 구문분석 시스템을 목표로 한다. 따라서 그래프 기반 접근방법보다 트랜지션 기반 접근방법을 선택하였다. 그리고 한국어에 적합한 트랜지션 시스템 기반 의존 파싱 모델을 제안한다.

한국어는 형태론적으로 교착어라는 특징을 가지고, 구문적으로 지배소 후위라는 특징을 가진다. 이는 영어권의 언어와는 다른 특징으로, 영어권에서 개발되고 활용되는 트랜지션 시스템에 지배소 후위라는 제약을 추가하면 더욱 효율적으로 구문분석을 수행할 수 있다. 예를 들어, 기존 트랜지션 기반 접근 방법은 3개에서 4개의 액션(action)을 정의하고 이를 분류한다. 하지만, 한국어의 지배소 후위 제약을 적용하면 이를 2개의 액션으로 단순화하고 더욱 효율적으로 구문분석을 수행할 수 있다. 그리고 본 논문에서는 트랜지션 기반 접근 방법의 가장 큰 단점인 에러 전파(error propagation)의 위험을

O: 교신저자

완화하기 위하여 백워드(backward) 기반 트랜지션 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존 한국어 구문분석 연구들을 살펴보고, 3절에서는 제안하는 트랜지션 시스템을 소개한다. 4절에서는 실험 및 결과를 보여주고, 마지막 5절에서 결론을 맺도록 한다.

2. 관련 연구

본 절에서는 한국어 의존 구문분석 관련 기존 연구를 살펴본다. 최근에 많이 연구된 한국어 의존 구분분석 방법으로는 영어권 연구와 비슷하게 그래프 기반 파싱 기법과 트랜지션 기반 파싱 기법이 있다.

그래프 기반 한국어 구문분석 연구로는 이용훈(2010), 임수종(2011), 안광모(2014)의 연구가 있다 [4-6]. 이용훈(2010)은 한국어의 지배소 후위 특징과 투사성 특징을 이용한 그래프 기반 파싱 알고리즘과 averaged perceptron 기법을 이용한 기계학습 모델을 제안하였다 [4]. 임수종(2011)은 그래프 기반 파싱을 수행하는데 자질의 가중치를 기계학습을 통하여 학습하여 높은 정확도를 보이는 파싱 기술을 제안하였다 [5]. 이용훈(2010)과 임수종(2011)은 모두 CYK 알고리즘에 기반하여 $O(n^3)$의 시간 복잡도를 보였다. 반면, 안광모(2014)는 한국어의 문법적 특징을 이용하여 각 의존소가 가질 수 있는 지배소 후보 집합을 필터링하고 이를 기반으로 파싱을 수행함으로써 효율성을 향상시켰다 [6].

트랜지션 기반 한국어 구문분석 연구로는 최진호(2011) 연구가 있다[7]. 최진호(2011)은 세종 구구조 코퍼스를 의존구조 코퍼스로 변환하는 방법을 제안하였고, 리스트에 기반한 트랜지션 시스템을 적용함으로써 높은 구문분석 성능을 보였다. 최진호(2011)의 연구는 본 논문의 연구와 가장 유사한 연구이다. 두 연구 사이의 차이는 본 연구는 한국어의 지배소 후위 특징을 트랜지션 시스템 자체에 반영하였고, 최진호(2011)는 일반 영어권의 연구에서 사용하던 트랜지션 시스템을 그대로 적용하였다는 점이다. 그리고 코퍼스 변환 방식에서도 일부 차이를 보였다. 최진호(2011)의 연구에서는 의존관계 레이블을 일반적인 "구문레이블_기능레이블" 조합이 아닌 PennTreeBank 스타일의 레이블로 변경하여 사용하였고, 대등절 연결 의존구조 태깅 가이드라인에 있어서도 일부 차이를 보였다.

마지막으로 한국어에 특화된 구문분석 기술 연구로는 다단계 구단위화를 이용한 의존 파싱, 확률 모델을 이용한 의존 파싱이 있다 [8-10]. 오진영(2010), 오진영(2013)에서는 키어절 자질 및 비어휘 자질을 이용한 다단계 구 단위화 기반 파싱 기술을 제안하였다 [8-9]. 박정열(2013)에서는 한국어 구문분석을 위하여 대용량의 데이터로부터 추출한 격틀(case frame) 정보를 활용한 어휘화(lexicalized) 확률 모델을 제안하였다 [10].

지배소 후위 원칙은 기존 연구에서도 가장 많이 사용되는 한국어의 특징이다. 본 연구에서는 트랜지션 시스템의 제약조건으로 지배소 후위 원리를 적용하여 더욱 효율적으로 구문분석을 수행할 수 있는 트랜지션 시스템을 제안한다.

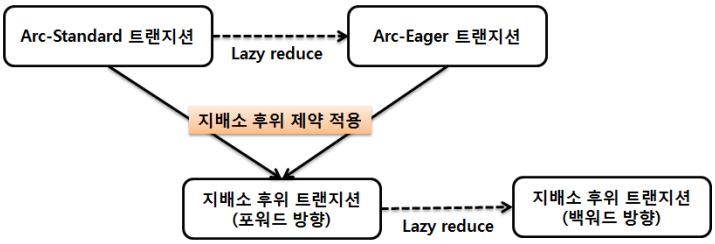

그림 1 트랜지션 시스템 간의 관계도

3. 지배소 후위 제약을 적용한 트랜지션 시스템

본 절에서는 한국어의 지배소 후위 제약을 적용한 트랜지션 시스템을 소개한다. 3.1절에서는 기존 arc-standard 및 arc-eager 트랜지션 기반 파싱 모델을 설명한다. 3.2절에서는 지배소 후위 제약을 적용한 파싱 모델을 소개하고, 3.3절에서는 arc-eager의 lazy-reduce를 적용한 백워드 기반 트랜지션 시스템을 소개한다. 3.4절에서는 트랜지션 시스템의 액션을 분류하기 위한 기계학습 기법 및 자질집합을 소개한다. 그림1은 본 논문에서 제안하는 트랜지션 시스템 사이의 관계를 설명한다.

3.1 트랜지션 시스템 기반 파싱 모델

본 연구에서는 의존 구문분석을 위하여 Nirve.(2008)의 트랜지션 시스템 정의을 이용한다[11]. 트랜지션 시스템 S의 정의는 아래와 같다.

$$S = (C, T, c_s, C_t)$$

1. C는 모든 가능한 컨피규레이션(configuration)의 집합이다. 각각의 컨피규레이션은 스택 σ, 버퍼 β,의존관계 집합 A를 포함한다.[1)]
2. T는 트랜지션 함수(transition function)의 집합이다. 트랜지션 함수 t는 C → C 로의 함수로 정의된다.
3. c_s는 초기화 함수이다. 입력 문장 x = (w_0, w_1, ..., w_n)를 특정 컨피규레이션 값 (스택 σ, 버퍼 β,의존관계 집합 A)으로 할당한다.
4. C_t는 종료(terminal) 조건을 정의한 컨피규레이션 집합으로, C의 부분집합으로 정의된다.

두 트랜지션 시스템의 구문분석 결과가 동일하더라도, 각 시스템에서 컨피규레이션, 트랜지션 함수, 초기화 함수, 종료 컨피규레이션을 어떻게 정의하느냐에 따라 서로 다른 처리 과정을 거쳐서 구문분석이 수행된다.

기존 트랜지션 시스템 중 가장 많이 사용되는 트랜지션 알고리즘으로는 arc-standard 와 arc-eager 트랜지션 알고리즘이 있다 [12-13]. 두 알고리즘은 기본적으로 영

1) 스택 σ는 사용하는 자료구조에 따라 리스트 λ로 표현되기도 한다.

$$\begin{aligned}
&\textit{Start state} && ([ROOT],[1,\dots,n],\{\}) \\
&\textit{Final state} && (S,[\,],A) \\
&\textbf{\textit{Transitions}} \\
&\text{Left}-\text{Arc}_l && := (\sigma|i\,,j|\beta,A) \Rightarrow (\sigma,j|\beta,A\cup\{(j,l,i)\}) \\
&\text{Right}-\text{Arc}_l^s && := (\sigma|i\,,j|\beta,A) \Rightarrow (\sigma,i|\beta,A\cup\{(i,l,j)\}) \\
&\textit{Shift} && := (\sigma,i|\beta,A) \Rightarrow (\sigma|i,\beta,A) \\
&\textbf{\textit{Preconditions}} \\
&\text{Left}-\text{Arc}_l && \neg[\,i=ROOT\,] \\
& && \neg\exists k\exists l'[\,(k,l',i)\in A\,] \\
&\text{Right}-\text{Arc}_l^s && \neg\exists k\exists l'[\,(k,l',j)\in A\,]
\end{aligned}$$

그림 3. Arc-standard 트랜지션

$$\begin{aligned}
&\textit{Start state} && ([ROOT],[1,\dots,n],\{\}) \\
&\textit{Final state} && (S,[\,],A) \\
&\textbf{\textit{Transitions}} \\
&\text{Left}-\text{Arc}_l && := (\sigma|i\,,j|\beta,A) \Rightarrow (\sigma,j|\beta,A\cup\{(j,l,i)\}) \\
&\text{Right}-\text{Arc}_l^e && := (\sigma|i\,,j|\beta,A) \Rightarrow (\sigma|i|j,\beta,A\cup\{(i,l,j)\}) \\
&\textit{Reduce} && := (\sigma|i,\beta,A) \Rightarrow (\sigma,\beta,A) \\
&\textit{Shift} && := (\sigma,i|\beta,A) \Rightarrow (\sigma|i,\beta,A) \\
&\textbf{\textit{Preconditions}} \\
&\text{Left}-\text{Arc}_l && \neg[\,i=ROOT\,] \\
& && \neg\exists k\exists l'[\,(k,l',i)\in A\,] \\
&\text{Right}-\text{Arc}_l^e && \neg\exists k\exists l'[\,(k,l',j)\in A\,] \\
&\textit{Reduce} && \exists k\exists l[\,(k,l,i)\in A\,]
\end{aligned}$$

그림 4. Arc-eager 트랜지션

어 구문분석을 위하여 개발되었고, 독일어, 스페인어, 체코어, 중국어와 같은 다양한 다중-언어(multi-lingual) 파싱에 효과적으로 적용된 방법론이다.

그림2는 arc-standard 트랜지션 알고리즘을 보여주고, 그림3은 arc-eager 트랜지션 알고리즘을 보여준다. 두 알고리즘은 공통적으로 문장의 왼쪽에서부터 오른쪽으로 진행하면서 의존관계가 있을 경우 아크(arc)를 생성하고, 의존관계가 없을 경우 shift(또는 reduce)를 수행한다는 공통점을 가진다. 차이점은 arc-standard 알고리즘은 right-arc 액션 시 오른쪽 노드를 바로 팝(pop) 하고, arc-eager 알고리즘은 오른쪽 노드를 바로 팝(pop) 하지 않고, 스택에 저장한 후 나중에 reduce 연산을 통하여 팝 한다는 점이다. (Lazy reduce) 즉, arc-standard 알고리즘은 두 노드 사이에 right-arc 관계가 있더라도 바로 right-arc를 수행하지 않고 오른쪽 노드에 대한 처리를 끝마친 이후에 right-arc 연산을 수행한다. 반면, arc-eager 알고리즘은 두 노드 사이에 right-arc 관계가 있다면 오른쪽 노드를 처리하기 이전에 먼저 right-arc를 수행한다. (그래서 arc-eager 라는

$$\begin{aligned}
&\textit{Start state} && ([\,],[1,\dots,n,ROOT],\{\}) \\
&\textit{Final state} && (S,[\,],A) \\
&\textbf{\textit{Transitions}} \\
&\text{Left}-\text{Arc}_l && := (\sigma|i\,,j|\beta,A) \Rightarrow (\sigma,j|\beta,A\cup\{(j,l,i)\}) \\
&\textit{Shift} && := (\sigma,i|\beta,A) \Rightarrow (\sigma|i,\beta,A) \\
&\textbf{\textit{Preconditions}} \\
&\text{Left}-\text{Arc}_l && \neg\exists k\exists l'[\,(k,l',i)\in A\,]
\end{aligned}$$

그림 2 지배소 후위 트랜지션 (포워드 방향)

명칭을 사용한다.)

3.2 지배소 후위 제약을 적용한 트랜지션 시스템

한국어는 교착어, 어순 자유 및 생략 가능, 지배소 후위라는 특징을 가지는 언어이다. 이 중, 지배소 후위(head final) 원리는 문장 내 각 어절의 지배소는 해당 지배소보다 뒤에 위치한다는 원리이다. 이는 이미 여러 구문분석 연구에서 많이 활용된 한국어의 주요한 특징이다.

지배소 후위 제약을 트랜지션 시스템에 적용하면, 문장 내 의존관계 중 right-arc가 존재하지 않게 된다. 따라서 기존 arc-standard 알고리즘에서 right-arc 연산을 제거할 수 있다. (arc-eager 알고리즘에서는 right-arc와 reduce 연산을 제거한다.) 문장 내 마지막 어절과 ROOT 역시 지배소 후위의 의존관계를 가지도록 ROOT의 위치를 문장의 뒤로 이동한다. (ROOT의 위치가 바뀜에 따라 기존 Left-Arc의 왼쪽 노드가 ROOT가 아니라는 제약조건도 제거된다.) 즉, 지배소 후위 제약을 적용한 최종적인 트랜지션 시스템은 그림 4와 같고, 이는 arc-standard와 arc-eager transition 모두 동일하다.

지배소 후위 원리는 문장 내 각 어절의 지배소는 자신보다 뒤에 위치한다는 것이기 때문에, 문장의 앞에 위치한 어절은 문장 전체가 지배소 후보가 되지만, 문장의 뒤에 위치한 어절은 자신보다 뒤의 어절만이 지배소 후보가 된다. 즉, 문장의 뒤에 위한 어절이 앞에 위치한 어절보다 오류를 범할 가능성이 더 적다고 유추할 수 있다.

트랜지션 기반 파싱의 가장 큰 단점은 처음 잘못된 트랜지션 액션을 결정하면, 그 결과가 계속해서 전파(propagation) 된다는 점이다. 즉, 전체 트랜지션 액션 열(action sequence) 중, 앞에서의 잘못된 결정이 뒤에서의 잘못된 결정보다 더 위험하다고 유추할 수 있다. 이 특징들을 종합하면, 트랜지션 시스템을 이용하여 한국어를 파싱할 경우 포워드 방식보다 백워드 방식이 유리하다는 점을 알 수 있다.

하지만, 그림 4와 같은 트랜지션 시스템으로는 백워드 방식의 파싱을 수행하기가 어렵다. 예를 들어, “백설공주가 예쁜 사과를 먹었다.” 문장의 경우, “사과를”과 “먹었다.” 사이를 먼저 트랜지션 액션을 결정해야 한다. 이 때, 둘 사이의 의존관계를 설정하고 “사과를”

$$\textit{Start state} \quad ([ROOT],[n,\dots,1],\{\})$$
$$\textit{Final state} \quad (S,[\,],A)$$

Transitions

$$\text{Left}-\text{Arc}_l^e \quad := (\sigma|i,j|\beta,A) \Rightarrow (\sigma|i|j,\beta,A\cup\{(i,l,j)\})$$
$$\textit{Reduce} \quad := (\sigma|i,\beta,A) \Rightarrow (\sigma,\beta,A)$$

Preconditions

$$\text{Left}-\text{Arc}_l^e \quad \neg\exists k\exists l'[(k,l',j)\in A]$$
$$\textit{Reduce} \quad \exists k\exists l[(k,l,i)\in A]$$

그림 5. 지배소 후위 트랜지션 (백워드 방향)

또는 "먹었다"를 팝(pop) 하면, "백설공주가" 또는 "예쁜"의 지배소(head)가 팝이 되는 것이어서 올바른 구문분석을 할 수 없다. 만약, 둘 사이의 의존관계를 설정하지 않고 나중에 판단한다면, 결국 문장의 앞에서부터 의존관계를 파악하는 포워드 방식과 같아진다.

3.3 Lazy-Reduce를 이용한 백워드 트랜지션 시스템

본 절에서는 arc-eager 트랜지션의 lazy-reduce 아이디어를 활용하여 지배소 후위 제약을 적용한 백워드 방식의 트랜지션 시스템을 소개한다.

제안하는 백워드 트랜지션 시스템에서는 입력 문장 x = (w_0, w_1, ..., w_n)를 x′ = (w_n, w_{n-1}, ..., w_1)과 같이 역순으로 변경한다. 그리고 arc-eager의 right-arc 연산을 적용하면 실제로는 left-arc가 생성되어 지배소 후위 제약이 유지된다. Right-arc 연산 수행 시, 양쪽 노드 모두 스택에 저장되기 때문에, shift 연산이 아닌 스택의 노드를 팝 하는 reduce 연산을 사용하다. 최종적인 백워드 트랜지션 시스템은 그림5와 같다.

마지막으로 [14]의 연구 결과를 보면, 트랜지션 시스템 기반 파싱은 주로 짧은 거리(short distance)의 의존관계에 강하고, 먼 거리(long distance)의 의존관계에 약하다는 특징을 가진다. 먼 거리 의존관계는 상대적으로 문장의 앞에서 자주 발생한다. 백워드 방식을 이용하면 먼 거리 의존관계를 파악할 때 현재까지 구축된 부분 트리(partial tree) 정보를 이용할 수 있다는 장점을 가진다.

예를 들어, "백설공주가 예쁜 사과를 먹었다."와 "백설공주가 먹은 사과는 독사과이다." 문장을 비교하면, 첫 문장의 "백설공주가"는 "먹었다."에 의존하고, 두 번째 문장의 "백설공주가"는 "먹은"에 의존한다. 만약 Forward 방식을 사용한다면 처음 "백설공주가-예쁜"만을 보고 shift를, "백설공주가-먹은"만을 보고 left-arc를 구분해야 하는 어려움이 있다. 하지만, 백워드 방식을 사용한다면, "예쁜 사과를 먹었다." 와 "먹은 사과는 독사과이다."에 해당하는 부분 구문분석(partial tree) 정보가 스택에 저장되어 있기 때문에 이를 이용할 수 있다.

3.4 기계학습 및 자질집합

표 1. 자질 집합

기본 자질 집합	- 노드: s, s_prev, s_next, s-1, s-2, s-3, q, q-1, q-2, q_prev 자질: f : {po, lm_po}, l : {po, lm_po}, f_l : po, l-1_l : {po, lm_po}, w : po - s_head:{f, l}:{lm_po}, s_head:w:po, s_head: label s_head2:{f, l}:{lm_po}, s_head2:w:po, s_head2: label - distance
확장 자질 집합	- 노드: (L,R) = {(s,q), (s-1,s), (s_prev,s), (s,s_next), (q,q-1), (q_prev,q)} 자질: { L:f_R:f, L:l_R:l, L:f_R:l,k L:l_R:f } : po { L:f_R:f, L:l_R:l, L:f_R:l, L:l_R:f } : lm_po L:(l-1_l)_R:(l-1_l) : { po, lm_po } L:f_L:l_R:f_R:l : po L:w_R:w : po - Combinations with distance - Combinations with head information - Combinations between above features
※ 노드 s: 스택 top, s-1: 스택 2번째 top, s-2: 스택 3번째 top, s_prev: 스택 top 앞 어절, s_next: 스택 top 뒷 어절 s_head: 스택 top의 head, s_head2: s_head의 head q: 버퍼 top, q-1: 버퍼 2번째 top, q-2: 버퍼 3번째 top q_prev: 버퍼 top 앞 어절 ※ 위치 f: 첫 형태소, w: 전체 형태소 l: 마지막 형태소, l-1: 뒤 2번째 형태소 ※ 속성 lm: lemma, po: pos, label: dependency label	

자연어 문장을 파싱하는 각 트랜지션 단계마다 어떤 액션을 선택할지 결정하기 위하여 기계학습 기법을 이용한다.

기계학습 알고리즘은 각 트랜지션 단계 마다 컨피규레이션에 해당하는 스택 σ, 버퍼 β,의존관계 집합 A를 입력으로 받는다. 출력으로는 Left-Arc와 Reduce (포워드 알고리즘의 경우 Shift)를 출력으로 사용한다. 이때, 의존관계의 레이블을 구분하기 위하여 Left-Arc에 대하여 Left-Arc:NP_SBJ 과 같이 레이블 별로 클래스를 세분화해서 사용한다. 이를 위해 자연어처리에서 많이 사용되는 최대 엔트로피 모델(Maximum entropy model)과 일반적으로 가장 좋은 성능을 보인다고 알려진 Structural SVM(Support Vector Machine)을 사용하였다 [15-16].

마지막으로, 본 연구에서 사용하는 자질 집합은 표1과 같다. 지배소 후위 제약을 적용한 포워드 및 백워드 트랜지션 시스템을 비교하기 위하여 표 1의 자질 집합을 같이 적용하였다.

4. 실험 환경 및 결과

4.1 실험 환경

표 2. 포워드-백워드 비교 실험 결과

		UAS	LAS	C-S
ME 기본자질	포워드	90.33	87.69	55.88
	백워드	91.11	88.56	59.11
	Δ	0.78	0.87	3.23
S-SVM 기본자질	포워드	91.29	89.04	58.93
	백워드	91.79	89.55	61.38
	Δ	0.5	0.51	2.45
S-SVM 확장자질	포워드	92.58	90.58	63.10
	백워드	92.85	90.82	64.46
	Δ	0.28	0.24	1.37

표 3. 자동 분석 형태소 기반 의존 파싱 실험 결과

	UAS	LAS	C-S
정답 형태소	92.85	90.82	64.46
자동 형태소	91.13	87.63	59.35
Δ	-1.72	-3.19	-5.11

본 연구에서는 실험을 위하여 세종 구구조 구문분석 코퍼스를 의존구조로 변환하여 사용하였다 [17]. 의존구조로의 변환은 한국어에서 일반적인 지배소 후위 규칙(head-right rule)을 적용하여 변환하였다. 후처리로 한국어의 특징을 반영하여 본용언-보조용언과 같은 동사구의 경우 의존소들이 보조용언에 걸리도록 수정하였다. 그리고 QA 및 정보검색 시스템에 실제적인 활용을 위하여 기호로 인한 어절 분리를 제거한 후 실험하였다. 예를 들어, < '백설공주가 먹은 사과' 는 독사과이다.> 문장의 경우 세종 구구조 코퍼스는 기호를 별도의 <사과' 는>을 < 사과 / ' / 는 > 의 3단위로 구분하였으나, 본 연구에서는 이를 하나의 단위로 합친 후 실험하였다. 마지막으로 세종코퍼스에서 1어절 문장은 실험에서 제외하였다. 최종적으로 실험에 사용한 세종코퍼스는 총 73,067 문장이고, 학습에 65,759 문장(90%), 테스트에 7,308 문장(10%)을 사용하였다.

시스템 평가 방법으로는 UAS(Unlabeled attachment score), LAS(Labeled attachment score), C-S (Complete match score)를 사용하였다. 파라미터 추정을 위하여 최대 엔트로피 모델은 L-BFGS 알고리즘을 사용하였고, Structural SVM은 Pegasos 알고리즘을 사용하였다 [18].

4.2 실험 결과

기본 자질 집합을 이용한 지배소 후위 트랜지션 시스템의 실험 결과는 표2와 같다. 기계학습 기법으로는 ME와S-SVM을 실험하였고, 자질 집합으로는 기본 자질집합과 확장 자질 집합을 실험하였다.

실험결과를 통해 최대 엔트로피 ahepfdf 사용하였을 때 본 논문에서 제안한 백워드 방식의 트랜지션 시스템이 포워드 방식보다 UAS 0.78%, LAS 0.87%, C-S 3.23% 정도 높은 성능을 보임을 알 수 있다. Structural SVM을 사용하였을 경우에도 백워드 방식이 포워드 방식보다 높은 성능을 보였지만, 성능 상승폭은 다소 줄어들었다. 이는 포워드 방식의 성능이 낮기 때문에 S-SVM으로 인한 분류 성능 향상이 에 더 많은 영향을 미친 것으로 사료된다. 마지막으로 본 연구에서 최고 성능을 보인 S-SVM 알고리즘에 확장자질 집합을 사용한 경우에도 백워드 방식이 포워드 방식보다 높은 성능을 보임을 알 수 있다. S-SVM에 확장 자질 집합을 사용한 경우의 성능은 UAS 92.85%, LAS 90.82%, C-S 64.46%이다.

표 4. 기존 연구 성능

	코퍼스	UAS	LAS	C-S
이용훈(2010)	국어정보베이스 (12,084문장)	88.42		35.13
오진영(2013)	세종코퍼스 (61,533문장)	85.61		43.48
오진영(2010)	세종코퍼스 (96,412문장)	87.03		42.88
임수종(2011)	세종코퍼스 (57,000여 문장)	88.15		
안광모(2014)	세종코퍼스 (69,429문장)	87.52		34.43
최진호(2011)	세종코퍼스 (56,063문장)	85.47	83.74	
박정열(2013)	세종코퍼스	86.43		

표 3은 형태소 분석 결과로 정답 형태소를 사용한 경우와 자동 분석 형태소를 사용한 경우의 실험 결과를 보여준다. 실험 환경은 S-SVM 알고리즘과 확장 자질 집합을 사용하였다. 자동 분석 형태소 실험은 세종 코퍼스에서 학습한 형태소 분석기를 이용하였고, 자동 형태소 분석기의 성능은 세종코퍼스 기준 96.3%이다. 제안하는 방법은 자동 분석된 형태소를 사용하여도 UAS 91.13%, LAS 87.63%의 높은 성능을 보임을 알 수 있다.

마지막으로 표 4에 기존 연구들의 실험 결과를 정리하였다. 많은 기존 연구들이 세종 코퍼스를 사용하였지만 사용한 코퍼스의 문장 수, 의존 구조 변환 규칙, 학습/실험집합 분리 기준 등이 서로 달라서 기존 실험 결과들을 직접 비교 하기는 어렵다. 다만, 본 연구 결과는 자동 형태소 분석 결과를 사용하더라도 90% 이상의 높은 성능을 보였고, 이는 QA 및 지식추출과 같은 실제적인 응용분야에 적용이 가능한 수준으로 고려된다.

5. 결론

본 논문에서는 한국어의 지배소 후위 제약을 트랜지션 시스템에 반영하여 arc-standard 알고리즘과 유사한 포워드 트랜지션 시스템과 arc-eager의 lazy reduce 방식을 이용한 백워드 트랜지션 시스템을 제안하였다. 실험결과 백워드 기반 트랜지션 시스템이 포워드 방식보다 높은 성능을 보였다. 이는 지배소 후위 원칙에 따라 뒷어절을 먼저 분석하는 것이 오류 가능성을 줄임으로써 트랜지션 시스템의 에러 전파(error propagation)를 방

지하는데 유리하고, 앞 어절을 분석할 때 현재까지 수행된 부분 구문분석(partial tree) 정보를 활용할 수 있다는 장점 때문으로 고려된다. 제안하는 트랜지션 시스템은 효율적으로 의존 파싱을 수행하면서도 UAS 92.8%의 높은 성능을 보였고, 자동 분석된 형태소를 이용할 경우에도 UAS 91% 이상의 성능을 보였다.

기존 연구들과의 비교를 위하여 기존 연구들을 조사하였지만, 서로 다른 코퍼스 양, 변환 방법, 분리 기준과 같은 이유로 직접적인 비교는 어려웠다. 하지만 절대적인 수치만을 비교하면 제안하는 방법이 기존 연구들에 뒤지지 않는 성능을 보임을 알 수 있었다. 그리고 기존 연구들은 주로 레이블이 없는 의존 구문분석을 연구하였지만, 본 연구에서는 의존 관계 레이블까지 분석하여 LAS 90% 이상의 높은 성능을 보였다.

향후 연구로는 최근 의존 구문분석에서는 트랜지션 접근 방법과 그래프 접근 방법을 하나의 모델로 통합하는 연구, 트랜지션 열 전체를 최적화하는 연구, 다중 언어를 위한 파싱 등이 주요한 연구 주제로 연구되고 있다. 한국어에서도 이와 같은 통합 모델 및 전역 최적화 기술에 대해 연구를 수행하면 더 높은 성능의 의존 구문분석이 가능할 것이다.

감사의 글

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10044577, (1세부) 휴먼 지식증강 서비스를 위한 지능진화형 WiseQA 플랫폼 기술 개발]

참고문헌

[1] S. Bucholz, E. Marsi, "CoNLL-X shared task on Multilingual Dependency Parsing," Proc. of CoNLL, pp.149-164, 2006.

[2] R. McDonald, K. Crammar, F. Pereira, "Online Large-margin Training of Dependency Parsers," Proc. of ACL, pp.91-98, 2005.

[3] J. Nivre, "An Efficient Algorithm for Projective Dependency Parsing," Proc. of IWPT, pp.149-160, 2003.

[4] 이용훈, 이종혁, "온라인 학습을 이용한 한국어 의존구문분석", 한국정보과학회 2010 한국컴퓨터종합학술대회 논문집 제37권 제1호(C), 2010.6, 299-304

[5] 임수종, 김영태, 나동열, "자질 가중치의 기계학습에 기반한 한국어 의존파싱", 정보과학회논문지, 소프트웨어 및 응용 제38권 제4호, 2011.4, 214-223

[6] 안광모, 서영훈, "지배소 후보 집합을 이용한 한국어 의존 구문 분석 알고리즘", 정보과학회논문지, 소프트웨어 및 응용 제 41 권 제 1 호(2014.1)

[7] J.D. Choi, Martha Palmer, "Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing", Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011), pages 1-11, Dublin, Ireland, October 6, 2011.

[8] 오진영, 차정원, "고성능 비어휘정보 한국어 구문분석", 2010 한국컴퓨터종합학술대회 논문집, Vol. 37, No. 1(C), 295-298

[9] 오진영, 차정원, "키어절을 이용한 새로운 한국어 구문분석", 정보과학회논문지: 소프트웨어 및 응용 제 40 권 제 10 호(2013.10), pp. 600-608

[10] Jungyeul Park, Daisuke Kawahara Sadao Kurohashi, Key-Sun Choi, "Towards Fully Lexicalized Dependency Parsing for Korean", iwpt2013

[11] Joakim Nivre, Algorithms for deterministic incremental dependency parsing, Computational Linguistics, v.34 n.4, p.513-553, December 2008

[12] Hiroyasu Yamada, Yuji Matsumoto, “Statistical Dependency Analysis With Support Vector Machines”, In proceedings of 8th International Workshop on Parsing Technologies, p. 195-206, 2003,

[13] J. Nivre, J. Hall, and J. Nilsson. “MaltParser: A data-driven parser-generator for dependency parsing.” In Proc. of LREC, 2006.

[14] Joakim Nivre, and Ryan McDonald, “Integrating Graph-Based and Transition-Based Dependency Parsers.” Proceedings of ACL-08: HLT, pages 950-958, 2008.

[15] Robert Malouf, “A comparison of algorithms for maximum entropy parameter estimation”, COLING-02 proceedings of the 6th conference on Natural language learning - Volume 20, Pages 1-7, 2002.

[16] Ioannis Tsochantaridis, et al, “Large Margin Methods for Structured and Interdependent Output Variables”, JMLR, Vol. 6, pages 1453-1484. 2005.

[17] 국립국어원, “21세기세종계획”, 2012

[18] S. Shalev-Shwartz et al., “Pegasos: Primal Estimated Sub-Gradient Solver for SVM,” Proc. ICML, Corvallis, Oregon, USA, June 20-24, 2007, pp. 807-814.